An outline of Qucs-S compact device modelling: History and capabilities: Part 1: Equation-Defined Device (EDD) modelling to Verilog-A module synthesis

Mike Brinson [1], mbrin72043@yahoo.co.uk.
Vadim Kuznetsov [2], ra3xdh@gmail.com

[1]Centre for Communications Technology, London Metropolitan University, UK

[2]Bauman Moscow Technical University, Russia

Qucs-S Internet facilities: documentation and software download sites

Qucs-S: Qucs with SPICE https://ra3xdh.github.io/

Qucs-S: Qucs with SPICE

Download links

The latest stable release is Qucs-0.0.19S. It is based on stable Qucs-0.0.19:

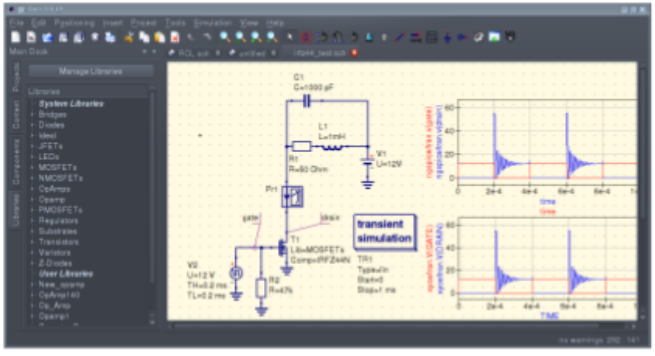
- Documentation:https://qucs-help.readthedocs.io/en/spice4qucs/
- Source tarball: qucs-0.0.19S.tar.gz
- Debian repository (32 and 64 bit packages), buit with openSUSE OBS:
  - Debian 8 "Jessy" at: download.opensuse.org
  - Debian 7 "Wheezy" at: download.opensuse.org
- Windows installer (Zipped EXE): qucs-0.0.19S-setup.zip

(Installation instructions...)

News

- *January, 26, 2017* Qucs-S 0.0.19 is released! The first stable release. Release announcement
- *November 8, 2016* Qucs-S RC8 released. Release notes and download link
- *September 3, 2016* Qucs-S RC7 released. Release notes and download link
- *May 15, 2016* Qucs-S RC6 released. Release notes and download link
- *March 23, 2016* Qucs-S RC5 released. Release notes and download link
- *January 31, 2016* Qucs-S RC4 released. Release notes and download link
- *August 29, 2015* Qucs-S RC3 released.
- *July 28, 2015* Qucs-S RC2 released.
- *July 25, 2015*Qucs-S RC1 released.

Simulation example with Qucs-S and Ngspice



(More screenshots...)

What is Qucs-S?

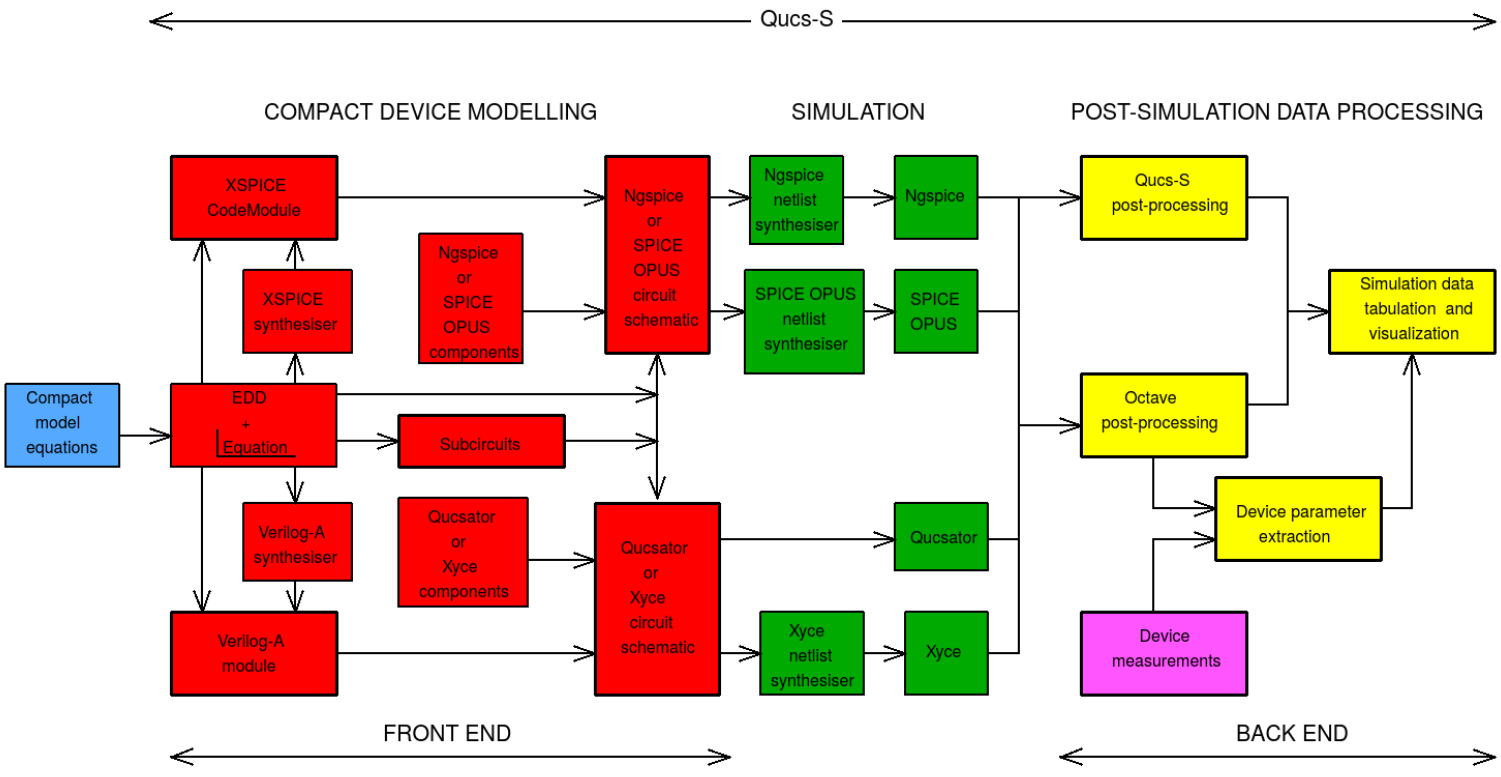
Qucs-S is a spin-off of the Qucs cross-platform circuit simulator. "S" letter indicates SPICE. The purpose of the Qucs-S subproject is to use free SPICE circuit simulation kernels with the Qucs GUI. It merges the power of SPICE and the simplicity of the Qucs GUI. Qucs intentionally uses its own SPICE incompatible simulation kernel Qucsator. It has advanced RF and AC domain simulation features, but the most of existing industrial SPICE models are incompatible with it. Qucs-S is not a simulator by itself, but it requires to use a simulation backend with it. The schematic document format of Qucs and Qucs-S are fully compatible. Qucs-S allows to use the following simulation kernels with it:

- Ngspice is recommended to use. Ngspice is powerful mixed-level/mixed-signal circuit simulator. The most of industrial SPICE models are compatible with Ngspice. It has an excellent performance for time-domain simulation of switching circuits and powerful postprocessor.
- XYCE is a new SPICE-compatible circuit simulator written by Sandia from the scratch. It supports basic SPICE simulation types and has an advances RF simulation features such as Harmonic balance simulation.
- SpiceOpus is developed by the Faculty of Electrical Engineering of the Ljubljana University. It based on the SPICE-3f5 code
- Qucsator as backward compatible

History of Qucs-S compact device modelling

- 21 Feb 2007 : Implementation of subcircuit parameters. Allow equation variables and sweep parameters in component properties and subcircuit parameters. Equations can be placed in subcircuits.
- 24 Feb 2007 : Input parameters of components can be used in Qucs-S equations.
- 21 Apr 2007 : Support for symbolically defined devices (EDD). ONLY explicit equations allowed.
- 2 Sep 2007 : Allow number engineering notation in equations (pre- and post-processing as well as EDD).
- Oct 2007 : Using ADMS to translate Verilog-AMS device models into C++ code. Manual compiling and linking of model C++ CODE.
- 2008 to 2011 : Generation of Verilog-A compact semiconductor device models.
- 3 Mar 2011 : Implementation of interactive GNU Octave connection to Qucs-S.
- 31 Aug 2014 : Dynamic compilation and loading of Verilog-A modules, Addition of a full ADMS/Qucs "turn key" Verilog-A compact device modelling system to Qucs. Users are no longer required to manually edit C++ code and build system to be able to run Verilog-A models. Uses ADMS 2.3.4.
- 23 Nov 2014 : Synthesis/translation of Qucs-S schematics/netlists to SPICE style netlists.
- 2014 to 2015 : Full set of SPICE commands added, for example .PARAM, and .OPTION.
- 2014 to 2015 : Full set of SPICE components added.
- 24 Aug 2015 : Verilog-A "Turn-Key" module synthesiser added.
- 28 Sept 2015 : Qucs PlotVs() function added to Qucs-S.
- 9 March 2016 : XSPICE distributed analogue device models and user defined Code Models added.
- 8 April 2016 : Addition of XSPICE Code Models in user generated model libraries.
- 4 Nov 2016 : Start developing XSPICE "Turn-Key" Code Model synthesiser.
- 1 Feb 2017 : EDD maximum number of branches increased from 8 to 20.
- 5 March 2017 : Added .FUNC and .include-script components.
- 2017 ..... : Continuing Qucs-S development in preparation for release 0.0.20.

A flow chart showing Qucs-S compact modelling facilities and data movement



NOTES:

1. Qucs-S allows the selection of the simulation engine to use.
2. Available simulation components depends on the simulation engine chosen.
3. Users may select either Qucs-S or Qctave post-processing of simulator data.

Building compact device models with Qucs-S: 1 Specification of the static and dynamic device properties of a semiconductor step recovery diode example

- In this presentation a compact model for a semiconductor step diode has been chosen to illustrate the different model building tools implemented by Qucs-S. This choice of model is deliberate because it's properties are well known, making the operation of the modelling tools easier to follow and understand.
- Non-linear static Id-Vd characteristics :
  $Id = IST2 \cdot (exp(Vd/(N \cdot Vt(T2)))) - 1.0) + GMIN \cdot Vd$, where
  $Vd = V(Anode, Cathode)$,
  $T1 = TNOM + 273.15$,
  $T2 = TEMP + 273.15$,
  $Vt(T2) = (k \cdot T2)/q$,
  $IST2 = IS \cdot AREA \cdot (T2/T1)^{XTN/N} \cdot exp(-Eg(300)/Vt(T2))$,
  $Eg(T) = Eg - (7.02e-4 \cdot T \cdot T)/(1108.0 + T)$, here
  k is the Boltzmann constant and q the elementary charge. Other physical parameters have their usual meaning: $AREA = 1, N = 1, IS = 1e-14$, $XTI = 3.0, Eg = 1.16, TNOM = 26.85, TEMP = 26.85$ and $GMIN = 1e-9$.

Building compact device models with Qucs-S: 1 Specification of the static and dynamic device properties of a semiconductor step recovery diode example

- Reverse breakdown voltage :
  $K2 = 1.0/(N \cdot Vt(T2))$, $K5 = N \cdot Vt(T2)$, $IBVEFF = IBV \cdot AREA$
  $IDBV = -IST2 \cdot (limexp(-BV \cdot K2) - 1.0)$,
  $BVEFF = (IBVEFF > IDBV)?BV - K5 \cdot ln(IBVEFF/IDBV) : BV$,
  $Id = -IST2 \cdot (limexp(-(BVEFF - Vd) \cdot K2) - 1.0 + BVEFF \cdot K2)$, where the breakdown physical parameters have their usual meaning: $BV = 4.5$, and $IBV = 1e-3$.
- Basic semiconductor diode depletion charge :
  $Qdep = (Vd >= 0.0)?CJ0T2 \cdot (Vd + P11 \cdot Vd \cdot Vd) : P6 \cdot (1 - (1 - Vd/JT2)^{P7})$,
  where
  $CJ0T2 = CJ0 \cdot AREA$, $P11 = M/(2 \cdot VJ)$, $P6 = (CJ0T2 \cdot VJT2)/P7$,
  $P7 = 1 - M$, and
  $VJT2 = (T2 \cdot VJ)/T1 - 2 \cdot Vt(T2) \cdot ln(T2/T1)^{1.5} - ((T2 \cdot Eg(T1)/T1) - Eg(T2)$,
  where the depletion capacitor physical parameters have their usual meaning:
  $CJ0 = 1e-12$, $VJ = 1.0$ and $M = 0.5$.

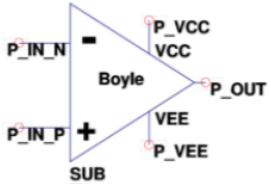
Building compact device models with Qucs-S: 1 Specification of the static and dynamic device properties of a semiconductor step recovery diode example

- Noise current :
  $\overline{i^2} = 2 \cdot q \cdot Id \cdot \Delta f + \frac{Kf \cdot Id^{Af}}{f} \cdot \Delta f + \frac{4 \cdot K \cdot T}{Rs} \cdot \Delta f$,
  where the noise physical parameters have their usual meaning:
  $Kf = 0.0, Af = 1.0$.
- Basic semiconductor diode diffusion charge :
  $Qdiff = TT \cdot Id$
- Step recovery diode charge :
  $Qd = (Vd <= 0.0)?CJ0 * Vd : 0.0$,
  $Qd = (Vd > 0.0)\&\&(Vd < FCP)?C1 * (Vd + C2)^2 - C3 : 0.0$,
  $Qd = (Vd > 0.0)\&\&(Vd > FCP)?Cf * Vd - Cf : 0.0$, where
  $FCP = FC * VJ, Cf = TAU/Rs, Cm = Cf - CJ0, C1 = Cf - CJ)/2 * FCP$,
  $C2 = (CJ0 * FCP)/Cn, C3 = (CJ0 * CJ0 * FCP)/(2 * Cm), C4 = Cm * FCP/2$,
  where the capacitor physical parameters have their usual meaning:
  $TAU = 2e - 9, Rs = 0.1, FC = 0.5$.

Qucs-S Equation-defined components - subcircuit/macromodel design equations

Qucs-S equation blocks can be used as a design aid to calculate component values at the start of a simulation sequence. Unlike Qucs the order of the equations in a Qucs-S block is important. Right hand variables must be calculated before use. Multiple blocks are combined by Qucs.

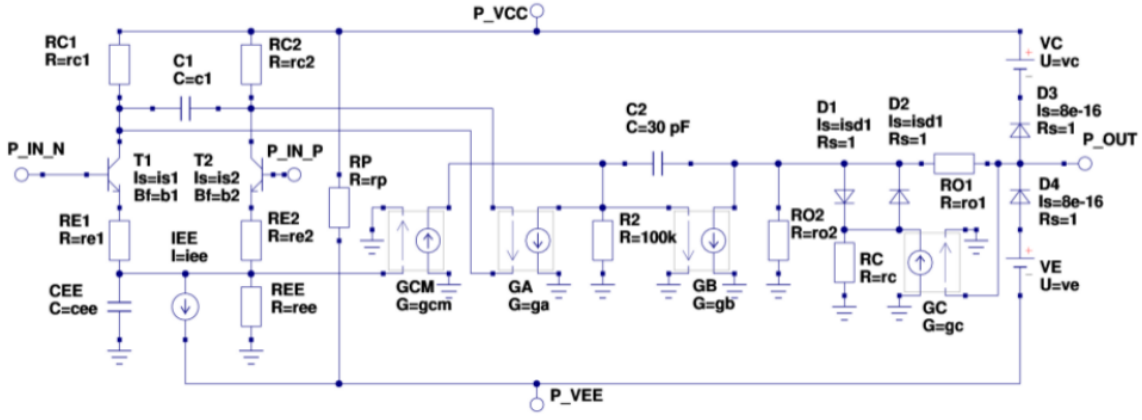
Adds a design element to subcircuits.



SUB
File=name
is=8e-16
vos=0.7e-3
c2=30e-12
r2=100e3
pswrt=0.625e6
nswrt=0.5e6
ios=20e-9
ib=80e-9
gbp=1e6
va=200
pm=70
cmrr=31622.8
avol=200e3
vsp=14.2
vsn=-13.5
vcc=15
vee=-15
iscp=25e-3
iscn=25e-3
pd=59.4e-3
ro1=76.8
ro2=489.2

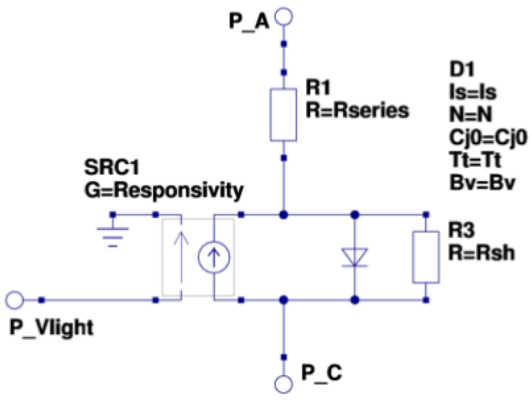
Equation
Eqn1
is1=is
ic1=0.5*c2*pswrt
ic2=ic1
VT=vt(300)
is2=is1*(1+vos/VT)
ib1=ib-0.5*ios
ib2=ib+0.5*ios
b1=ic1/ib1
b2=ic2/ib2
iee=ic1/(((b1+1)/b1)+((b2+1)/b2))
gm1=ic1/VT
rc1=1/(2*pi*gbp*c2)
rc2=rc1
re1=((b1+b2)/(2+b1+b2))*(rc1-1/gm1)
re2=re1
ree=va/iee
cee=(2*ic1/nswrt)-c2
dphi=90-pm
c1=0.5*c2*tan(dphi*pi/180)
gcm=1/(cmrr*rc1)
ga=1/rc1
gb=(avol*rc1)/(r2*ro2)
ix=2*ic1*r2*gb-is1
temp1=-(ro1*is1/VT)
isd1=ix*exp(temp1)+1e-32
temp2=ix/isd1
rc=VT*ln(temp2)/(100*ix)
gc=1/rc
vc=abs(vcc)-vsp+VT*ln(iscp/is1)
ve=abs(vee)-vsn+VT*ln(iscn/is1)
rp=(vcc-vee)*(vcc-vee)/pd

Boyle OP AMP macromodel

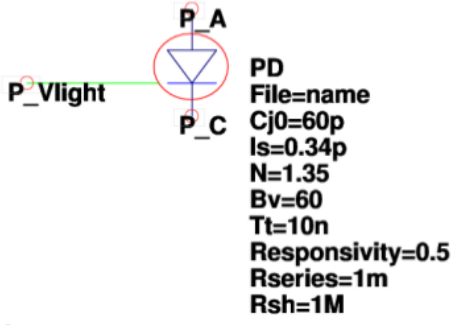


Qucs-S: Subcircuit / macromodels

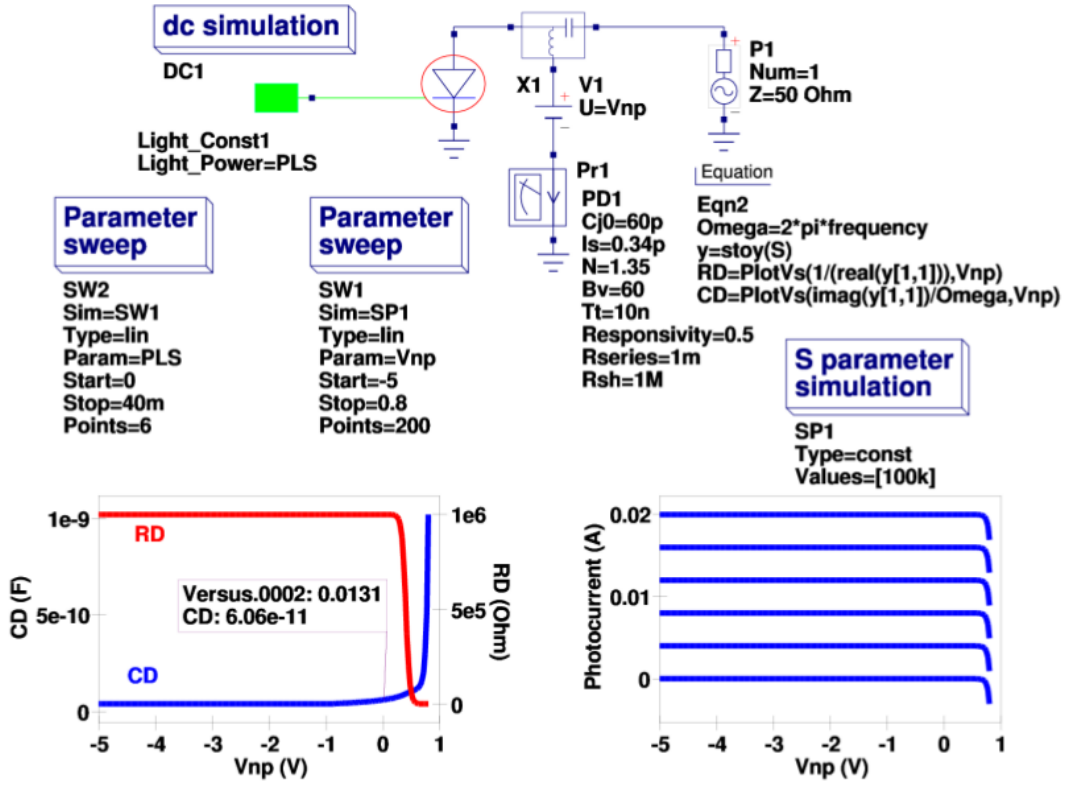
Photodiode subcircuit body



Photodiode symbol

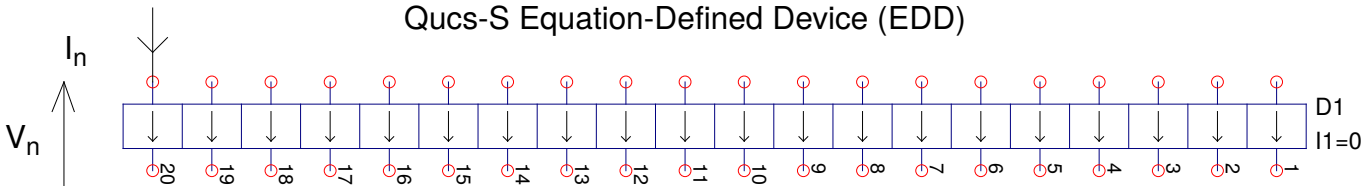


Test circuit and simulation data



Green denotes light source and light bus

Qucs-S: Nonlinear equation defined devices (EDD)



$I = I(V)$, $g = dI/dV$

$Q = Q(V,I)$, $C = dQ/dV = \partial Q(V)/\partial V + \partial Q(I)/\partial I \cdot g$, where

the current flowing in branch n is $I_n = I(V_n) + d/dt(Q_n)$, and $1 <= n <= 20$.

- EDD is a multiterminal nonlinear component with branch currents that can be functions of EDD branch voltage, and stored charge that can be a function of both EDD branch voltages and currents
- EDD is similar, but more advanced to the SPICE 3f5 B type I or V controlled sources
- EDD can be combined with conventional circuit components and Qucs-S equation blocks when constructing compact device models and subcircuit macromodels
- EDD is an advanced component, allowing users to construct prototype experimental models from a set of equations derived from physical device properties
- EDD operator d/dt is undertaken internally by Qucs-S
- Qucs-S EDD can have a maximum of 20 two terminal branches

Jahn S. & Brinson M.E. (2008). Interactive compact device modelling using Qucs equation-defined devices. International Journal of Numerical Modelling: Electrical Networks, Devices and Fields, 21:335-349, DOI:10.1002/jnm.676, John Wiley & Sons, Ltd.

Brinson M.E. & Jahn S. (2009), Qucs: A GPL software package for simulation, compact device modelling and circuit macromodelling from DC to RF and beyond, International Journal of Numerical Modelling: Electrical Networks, Devices and Fields, 22:297-319, DOI:10.1002/jnm.702, John Wiley & Sons, Ltd.

Qucs-S: An EDD compact model of a semiconductor diode, including noise

Model
Function
Control
Parameters

BVSWITCH
CdepSWITCH
CdiffSWITCH
ACnoiseSWITCH

Noise free resistor

Shot noise

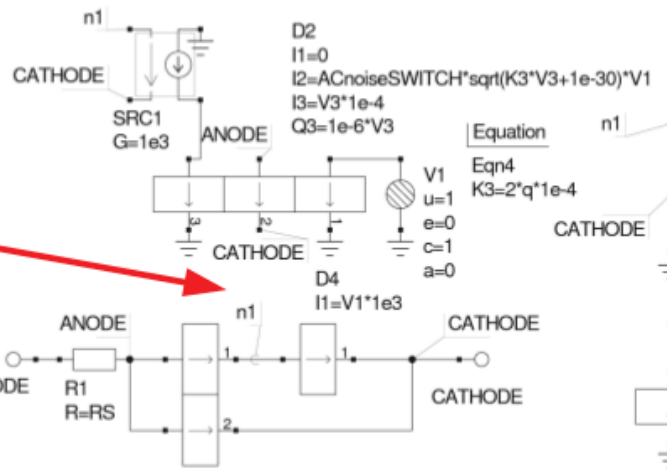
1/f noise

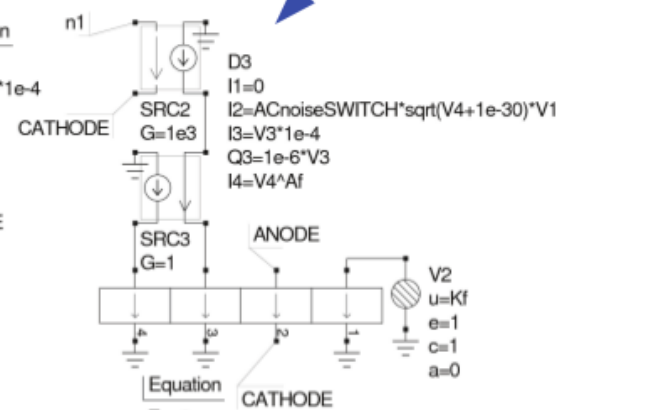
$Q1 = Qdep$

$Q2 = Qdiff$

I_d

Breakdown I_d





D1
I1= IST2*(limexp (V1*K2)-1.0)+GMIN*V1
Q1=(CdepSWITCH==1) ? ((V1 >=0.0) ? CJ0T2*(V1+P11*V1*V1) : P6*(1-(1-V1/VJT2)^P7)) : 0
I2=(BVSWITCH==1) ? (-IST2*(exp(-(BVEFF+V1)*K2)-1.0 +BVEFF*K2)) : 0
Q2=(CdiffSWITCH==1) ? TT*I1 : 0

Equation

Eqn3
P6=CJ0T2*VJT2/P7
P7=1-M
K5=N*VTT2
IBVEFF=IBV*AREA
IDBV=-IST2*(exp(-BV*K2)-1.0)
BVEFF=(IBVEFF>IDBV) ? BV-K5*ln(IBVEFF/IDBV) : BV
P11=M/(2*VJ)

Equation

Eqn1
GMIN=1e-12
ISEFF=IS*AREA
K2=1/(N*VTT2)
CJ0T2=CJ0*AREA

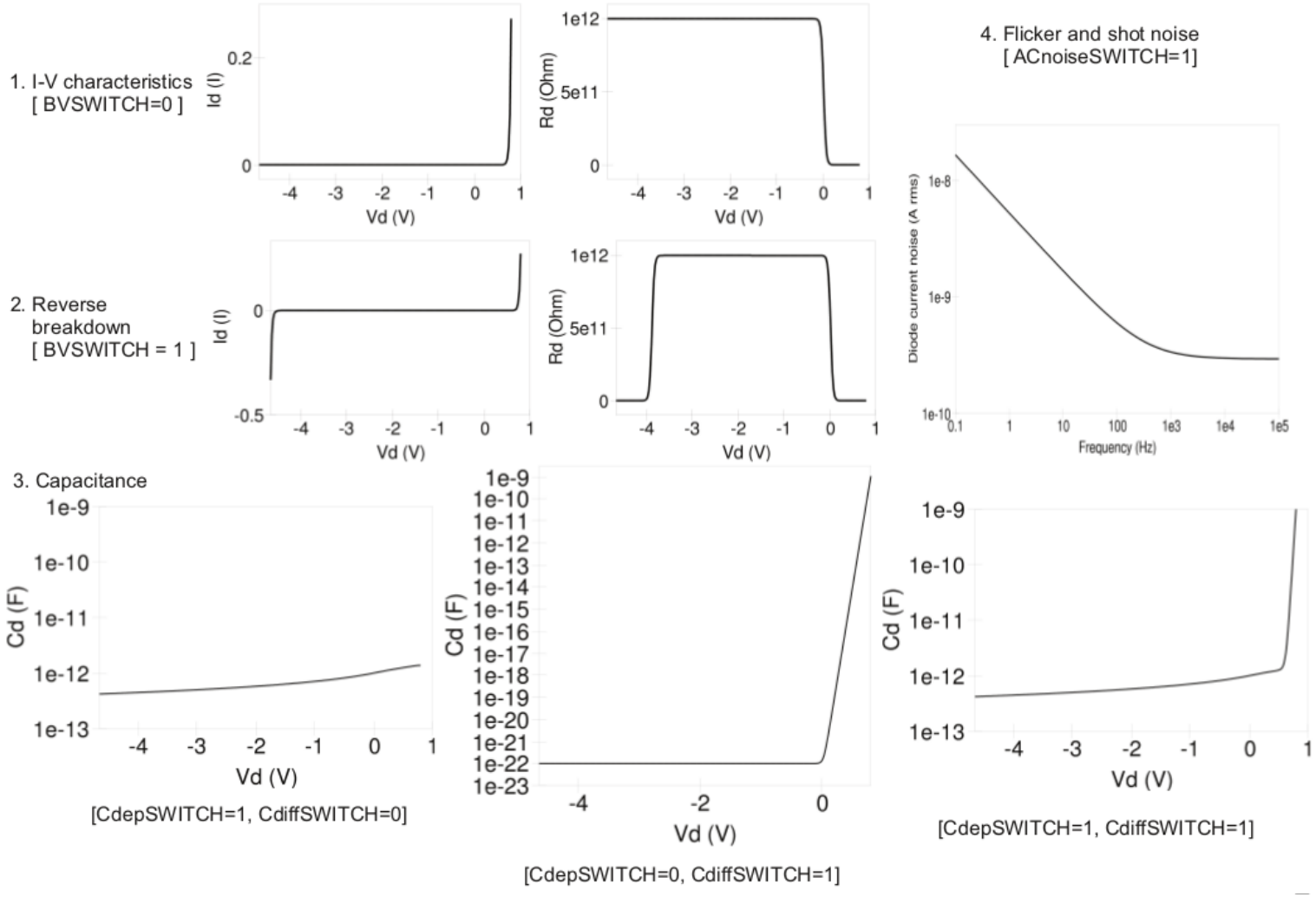
Equation

Eqn4
K3=2*q*1e-4

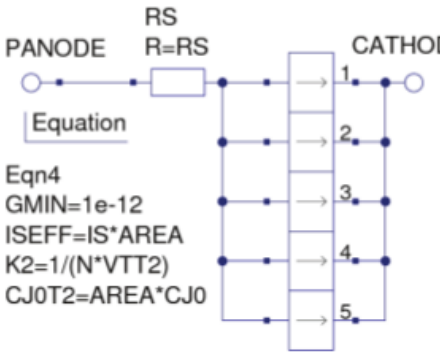
Equation

Eqn2
T1=TNOM+273.15
VJT2= (T2*VJ/T1)-2*VTT2*ln(T2/T1)^1.5-((T2*EgT1/T1)-EgT2)
VTT2=PK*T2/PQ
EgT1=Eg-7.02e-4*T1*T1/(1108+T1)
EgT2=Eg-7.02e-4*T2*T2/(1108+T2)
T2=TEMP+273.15
IST2= ISEFF*(T2/T1)^(XTI/N)*exp((-EgT1/VTT2)*(1-T2/T1))
PQ=1.602176462e-19
PK=1.3806503e-23

Qucs-S: An EDD compact model of a semiconductor diode, typical simulation data



Qucs-S: An EDD compact model of a step recovery semiconductor diode



Equation

Eqn4
GMIN=1e-12
ISEFF=IS*AREA
K2=1/(N*VTT2)
CJ0T2=AREA*CJ0

D1
I1= IST2*(limexp (V1*K2)-1.0)+GMIN*V1
Q1=(CdepPARAM==1) ? ((V1 >=0.0) ? CJ0T2*(V1+P11*V1*V1) : P6*(1-(1-V1/VJT2)^P7)) : 0
I2=(BVSWITCH==1) ? (V1 < -BV) ? -IST2*(limexp(-(BVEFF+V1)*K2) -1 +BVEFF*K2) : 0 : 0
Q2=(CdiffPARAM==1) ? TT*I1 : 0
I3=(BVSWITCH == 1) ? (V1 == -BV) ? IBV : 0 : 0
Q3=(IrecSWITCH == 1) ? (V1 <0.0) ? CJ0*V1 : 0 : 0
Q4=(IrecSWITCH == 1) ? (V1 > 0.0) && (V1 < FCP) ? C1*(V1+C2)^2-C3 : 0 : 0
Q5=(IrecSWITCH == 1) ? (V1 > FCP) ? CF*V1-C4 : 0 : 0

Q3 represents Cdep

Q4 and Q5 represent Cdiff

Equation

Eqn2
T1=TNOM+273.15
VJT2= (T2*VJ/T1)-2*VTT2*ln(T2/T1)^1.5-((T2*EgT1/T1)-EgT2)
VTT2=PK*T2/PQ
EgT1=Eg-7.02e-4*T1*T1/(1108+T1)
EgT2=Eg-7.02e-4*T2*T2/(1108+T2)
T2=TEMP+273.15
IST2= ISEFF*(T2/T1)^(XTI/N)*exp((-EgT1/VTT2)*(1-T2/T1))
PQ=1.602176462e-19
PK=1.3806503e-23

Equation

Eqn5
FCP=VJ
CF=TAU/RS
CM=CF-CJ0
C1=CF-CJ0/2*FCP
C2=(CJ0*FCP)/CM
C3=(CJ0*CJ0*FCP)/(2*CM)
C4=CM*FCP/2
CdepPARAM=(IrecSWITCH == 1) ? 0: (CdepSWITCH ==1) ? 1 : 0
CdiffPARAM=(IrecSWITCH == 1) ? 0: (CdiffSWITCH ==1) ? 1 : 0

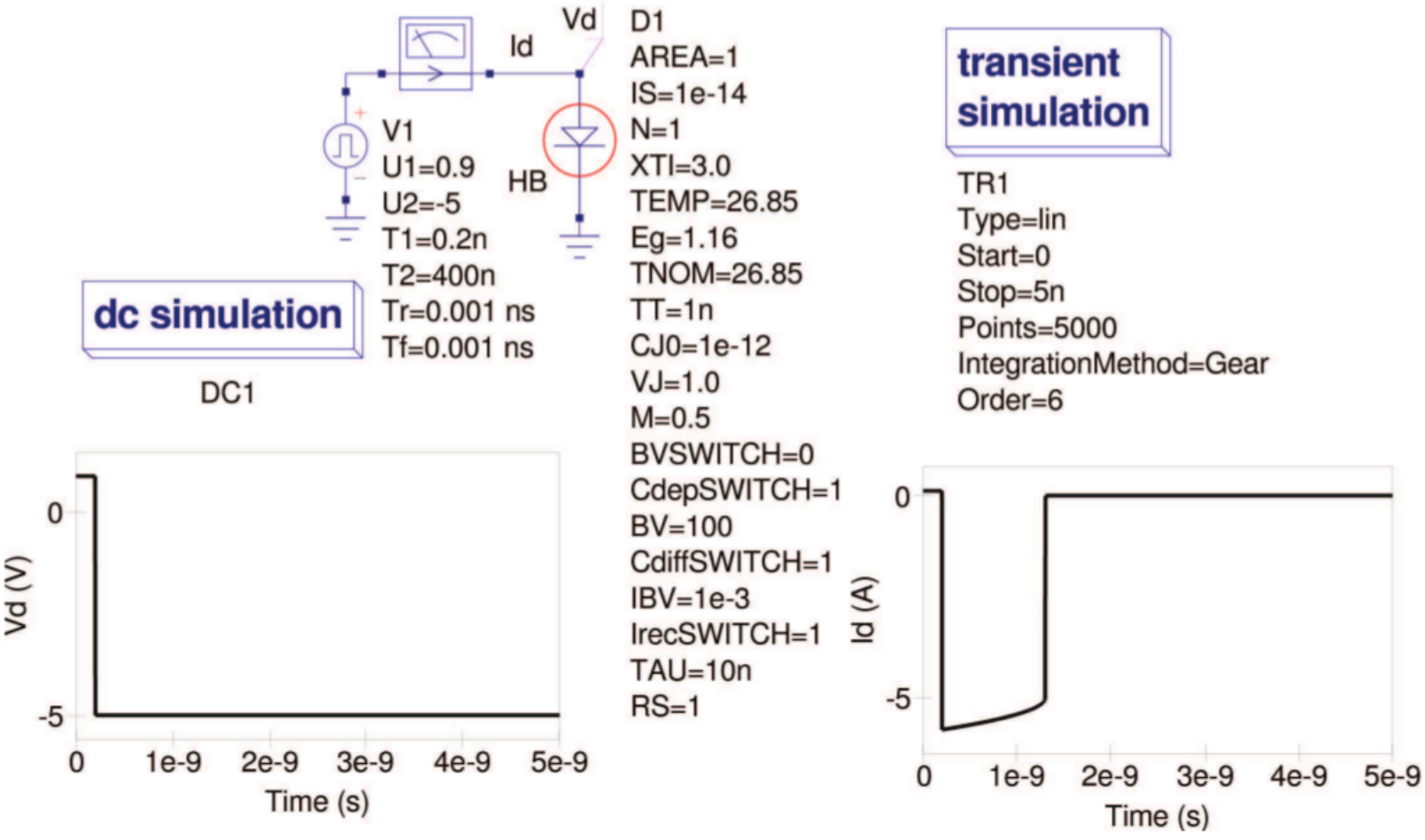
Equation

Eqn3
P6=CJ0T2*VJT2/P7
P7=1-M
K5=N*VTT2
IBVEFF=IBV*AREA
IDBV=-IST2*(limexp(-BV*K2)-1.0)
BVEFF=(IBVEFF>IDBV) ? BV-K5*ln(IBVEFF/IDBV) : BV
P11=M/(2*VJ)

If IrecSWITCH == 1 then CdepSWITCH and CdiffSWITCH over-riden

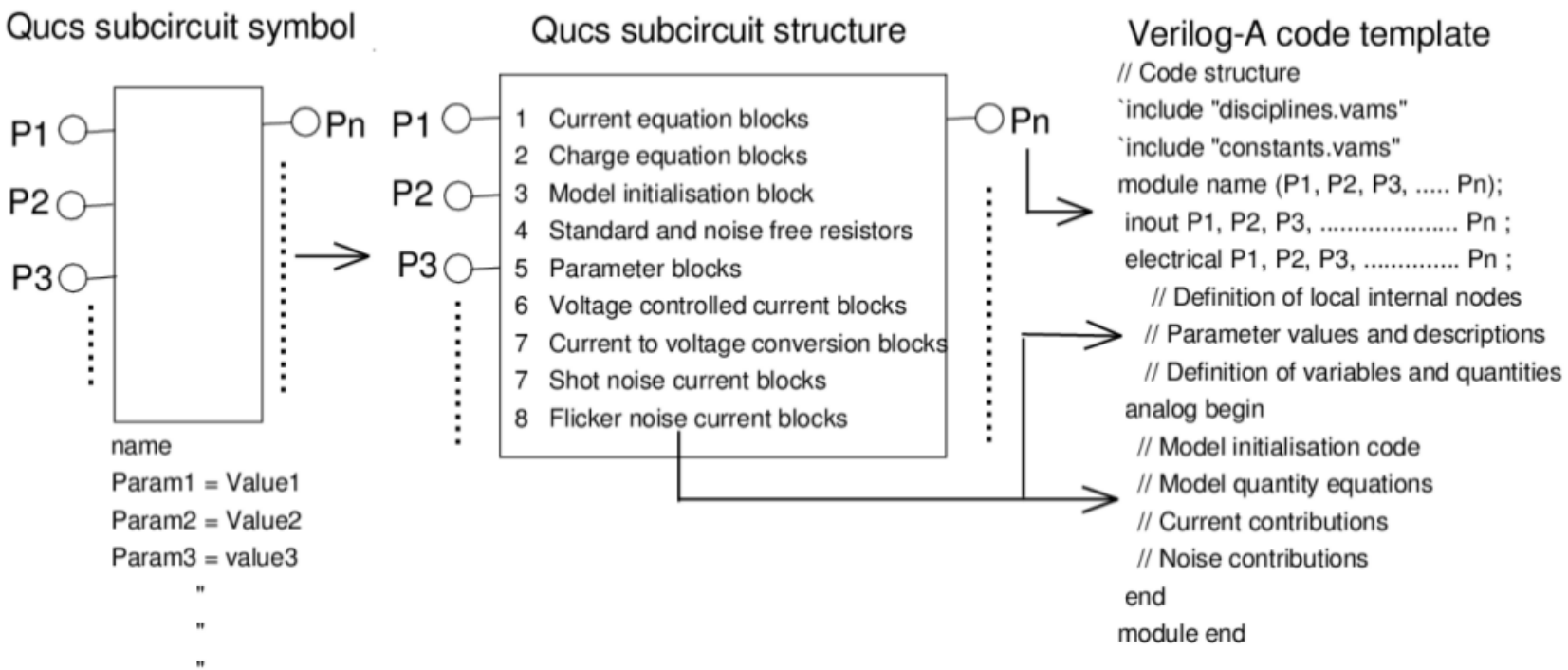
Small signal AC shot and 1/f noise model not included

Qucs-S: Test circuit and simulation data for the step recovery semiconductor diode model



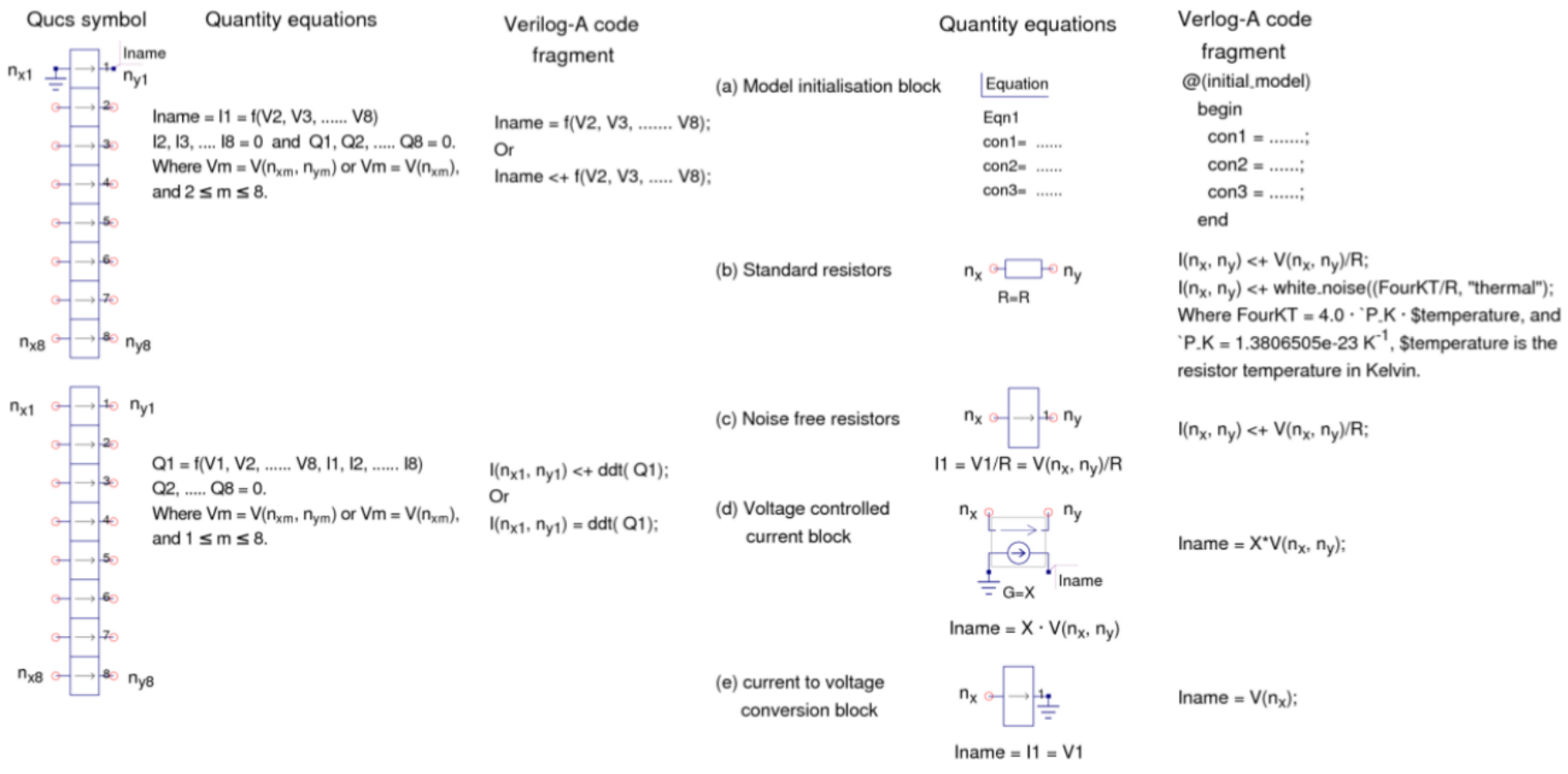
Generating Qucs-S Verilog-A compact device models: Introduction

- The following diagram illustrates the initial stage in the construction of a Qucs Verilog-A compact device model.



Relationships between Qucs-S schematic symbols and Verilog-A code fragments

Fundamental EDD blocks



A maximum of 20 two port branches are now allowed per EDD.

MOT-ADMS: Introduction to the basic Verilog-A subset available with ADMS

The MOT-ADMS software is supplied with little documentation!
These brief notes provide a basic introduction to the MOT-ADMS Verilog-A subset

- Verilog-A is a case sensitive language
- Comments: single line comments start with //,
  block comments begin with /* and end with */
- Identifiers are sequences of letters, digits, dollar signs '$' and the underscore '_'; the first letter of an identifier must not be a digit
- MOT-ADMS version 2.30 keywords: **parameter, aliasparameter, aliasparam, module, endmodule, function, endfunction, discipline, potential, flow, domain, ground, enddiscipline, nature, endnature, input, output, inout, branch, analog, begin, end, if, while, case, endcase, default, for, else, integer,real, string, from, exclude, inf, INF**
- Compiler directives: **`define,`undef, `ifdef, `else, `endif, `include**
- Data types: **integer**s, **real**s and **string**s
- Predefined constants in “constants.vams”: **`M_PI, `M_TWO_PI, `M_PI_2, `M_PI_4, `M_1_PI, `M_2_PI, `M_2_SQRTPI, `M_E, `M_LOG2E, `M_LOG10E, `M_LN2, `M_LN10, `M_SQRT2, `M_SQRT1_2, `P_Q, `P_C, `P_K, `P_H,`P_EPS0, `P_U0, `P_CELSIUS0**
- Variables are named objects that contain a value of a particular type. They are initialised to zero or unknown. They retain their value until changed by an assignment statement.

MOT-ADMS: Introduction to the basic Verilog-A subset available with ADMS; continued

- Parameters are declared using statements of the form:
  parameter integer size=16; parameter real period = 1.0 from (0:inf);
  parameter integer dir = 1 from [-1:1] exclude 0;
- Verilog-A natures and disciplines ae listed in file “disciplines.vams”
- Port, net and node examples in Verilog-A:

```
module amp(out1, in1);
  input in1;
  output out1;
  electrical out1, in1;
```

- Branches declared with statement **branch (n1,n2) b1;**
- Signal access function examples: **V(n2), I(n), V(b1), I(b1), V(n,m), I(n,m)**
- Current contribution examples:

```
I(diode) <+ Is*(limexp(V(diode)/$vt)-1);
I(diode) <+ ddt(-2*cj0*phi*sqrt(1-V(diode)/phi));
```

- MOT-ADMS allows an extensive range of Verilog-AMS operators and mathematical functions
- Environmental Functions: **$temperature, $vt, $strobe, $finish, $given, $parameter_given**
- Analogue operators: **@(initial_step), @(final_step), @(initial_model), @(initial_instance)**

MOT-ADMS: Introduction to the basic Verilog-A subset available with ADMS; continued

- Analogue behavioural statements:

1. Analog process/procedural block;

```
analog begin
    I(diode) <+ Is*(limexp(V(diode)/$vt)-1);
    qd       =  tf*I(diode) -2*cj0*phi*sqrt(1-V(diode)/phi);
    I(diode) <+ ddt(qd);
end
```

2. Assignment statements consist of a variable followed by = and an expression
3. Conditional operator cond ? Val1 : Val2, for example

```
State = (V(d) > 0.0) ? 1 : -1;
```

4. if-else statement:

```
If (V(ps,ns) > thresh)
    I(p,n) <+ 1;
else
    I(p,n) <+ 0;
```

TRUE = non-zero value

5. Case statement:

```
case (select)
  0 : out = I(in0);
  1 : out = I(in1);
  2 : out = I(in2);
  default : out = 0;
endcase
```

6. While statement:

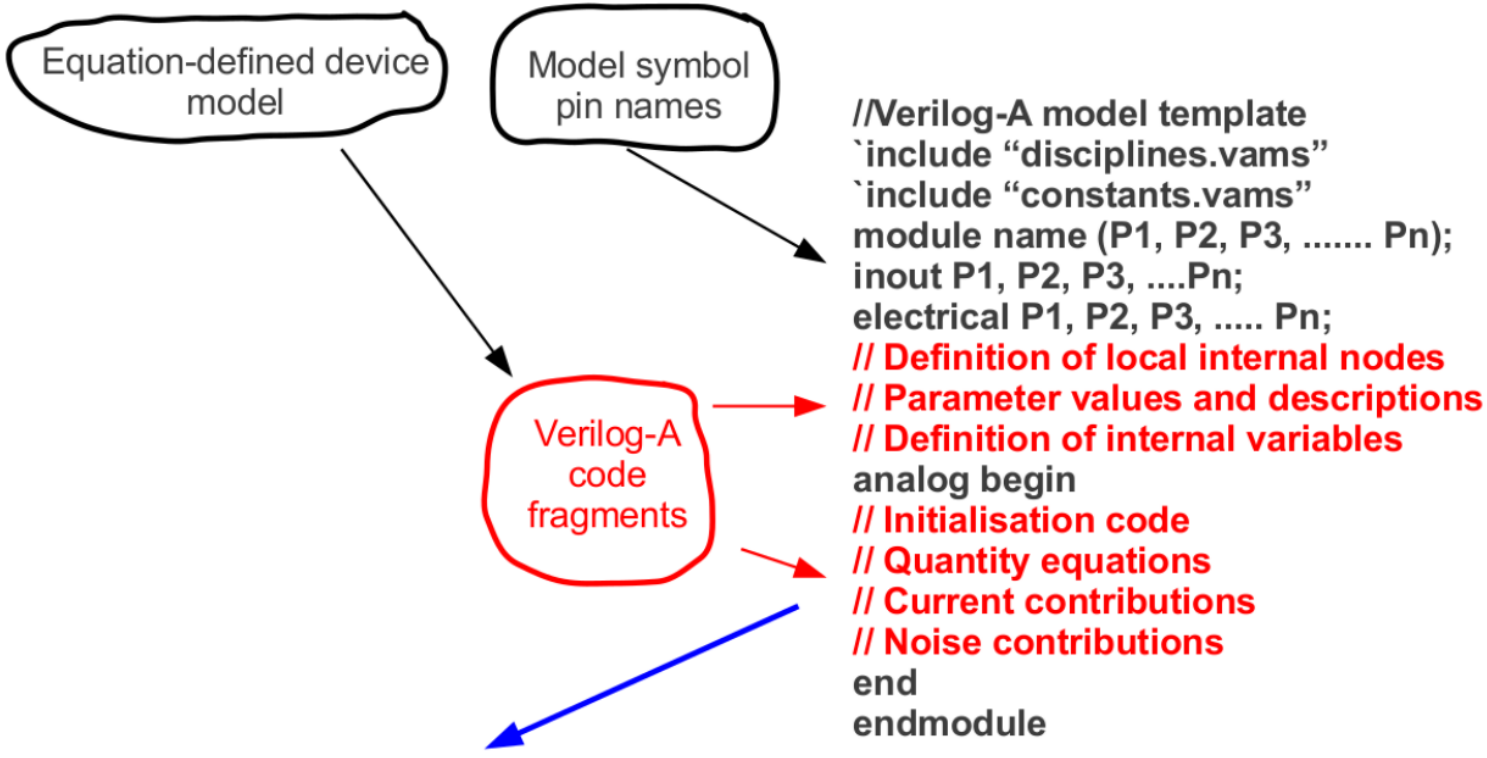
```
test = 4;
While( test) begin
  A = A+1;
  B = B-1;
  test = test-1;
end
```

7. For loops:

```
for (i=0; I <10; i=i+1) begin
                ..
                ..
end
```

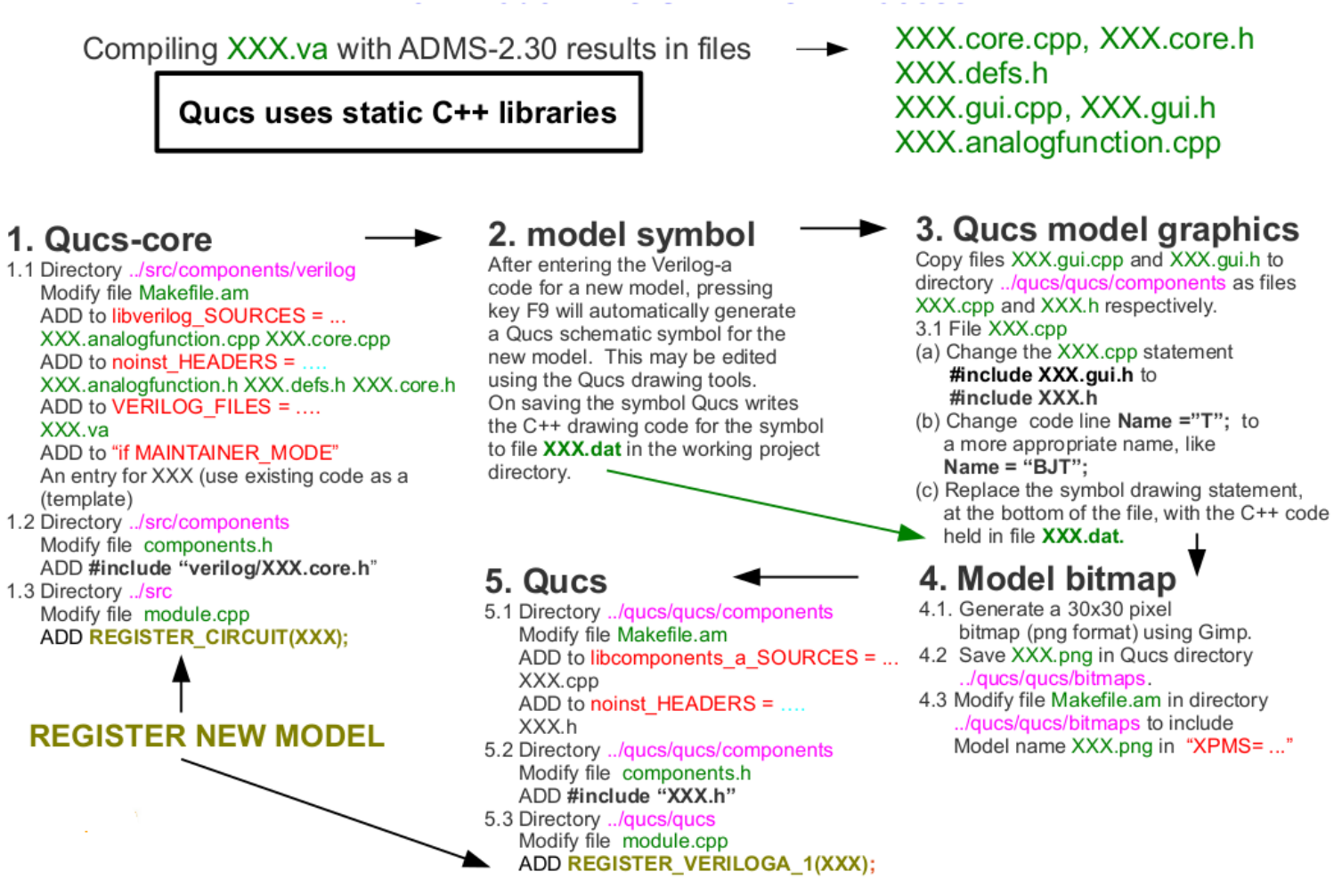
8. User defined functions and function calls.

Generating Qucs-S Verilog-A compact device models: original user controlled construction of Verilog-A models using static C libraries



1. Compile Verilog-A template code with ADMS
2. Add new model to Qucs by patching C++ code
3. Add new model symbol to Qucs C++ code
4. Compile and link Qucs **static** C++ code to generate new version of Qucs

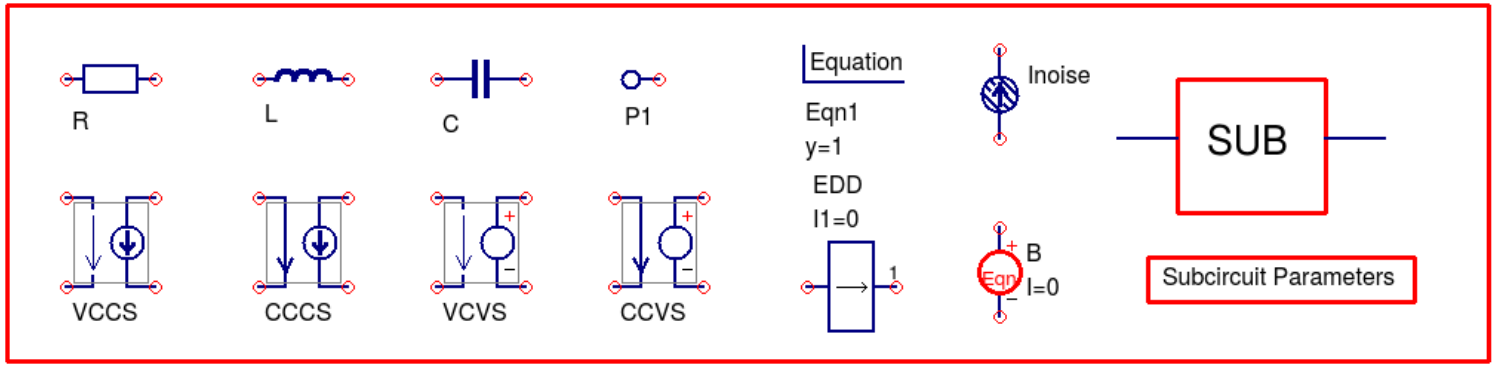
Generating Qucs Verilog-A compact device models: C++ code patches; model REGISTRATION process



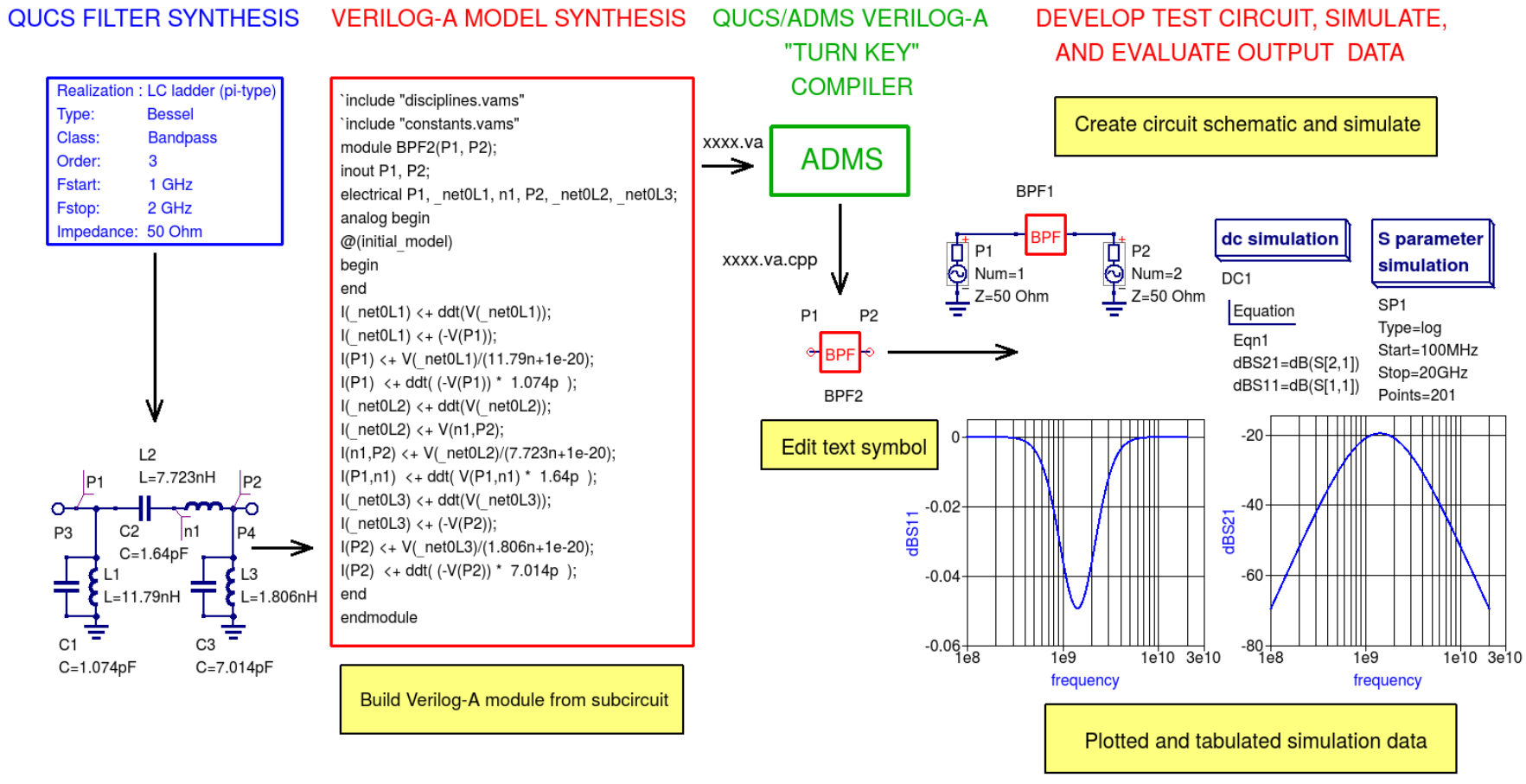
Introduction to the Qucs GPL Verilog-A module synthesizer: Part I

Qucs-0.0.19S includes the first release of a GPL Verilog-A synthesis tool for compact device modelling.

- The Qucs-0.0.19-S Verilog-A synthesizer is a basic working version of this new open source ECAD tool.
- Generated synthesized Verilog-A code is relatively basic and has to be optimized manually for speed. However, it is expected that in the future its operation will improve as development of the Qucs synthesizer progresses.
- Circuits and Verilog-A synthesized models can be constructed from the following Qucs/SPICE built in components:

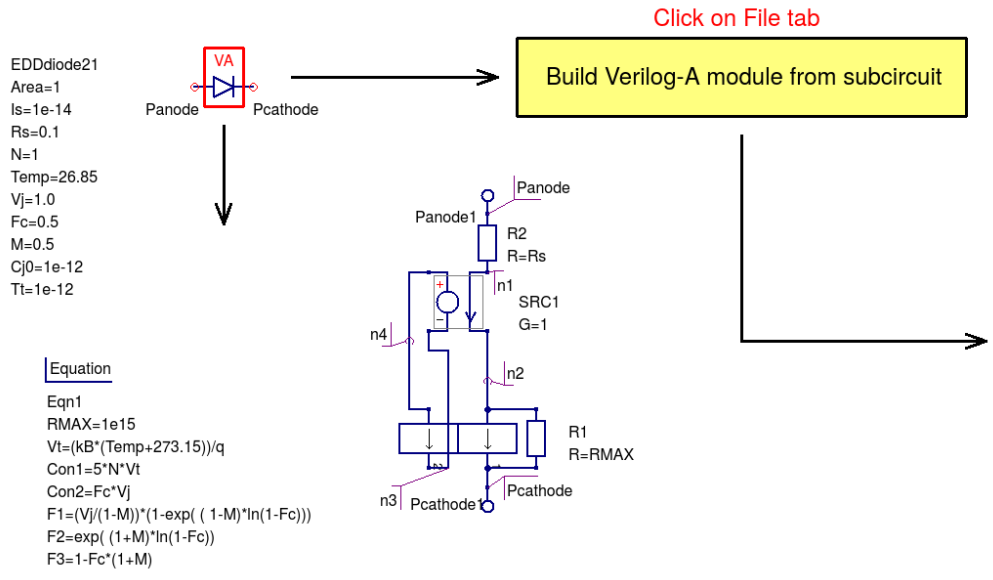


Data flow through the Qucs GPL compact device modelling tool set.



Introduction to the Qucs GPL Verilog-A module synthesizer: Part IV

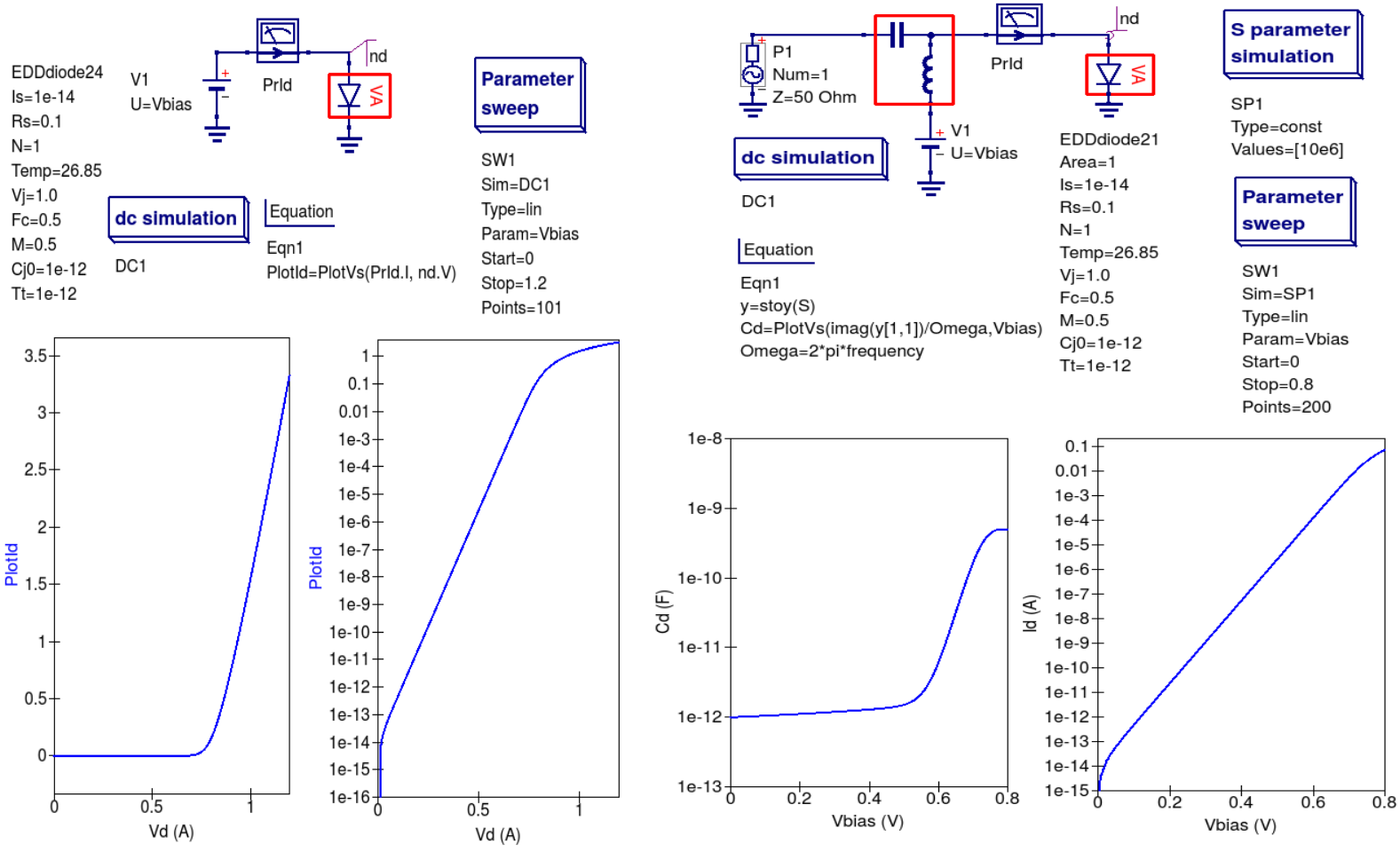
Synthesis of a SPICE like compact semiconductor diode model: static I_d and dynamic capacitance model plus synthesized Verilog-A module code.




```
`include "disciplines.vams"
`include "constants.vams"
module EDDdiode2(Panode, Pcathode);
inout Panode, Pcathode;
electrical Pcathode, n2, n1, Panode, n4, n3;
parameter real Area=1;          parameter real Is=1e-14;
parameter real Rs=0.1;          parameter real N=1;
parameter real Temp=26.85;      parameter real Vj=1.0;
parameter real Fc=0.5;          parameter real M=0.5;
parameter real Cj0=1e-12;       parameter real Tt=1e-12;
real RMAX, Vt, Con1, Con2, F1, F2, F3;
analog begin
@(initial_model)
begin
RMAX=1e15;     Vt=(`P_K*(Temp+273.15))/`P_Q;
Con1=5*N*Vt;   Con2=Fc*Vj;
F1=(Vj/(1-M))*(1-exp((1-M)*ln(1-Fc)));    F2=exp((1+M)*ln(1-Fc));
F3=1-Fc*(1+M);
end
I(Pcathode,n2) <+ V(Pcathode,n2)/( RMAX );
I(n1,Panode) <+ V(n1,Panode)/( Rs );
I(n1,Panode) <+ white_noise( 4.0*`P_K*( 26.85 + 273.15) / ( Rs ), "thermal" );
I(n1,n2) <+ V(n1,n2) * 1e3; I(n3,n4) <+ I(n3,n4) *1e-3;  I(n3,n4) <+ V(n1,n2) * 1e6 * 1 ;
I(n2,Pcathode) <+ Area*Is*(limexp(V(n2,Pcathode)/(N*Vt))-1);
I(n2,Pcathode) <+ ddt( (V(n2,Pcathode)<=Con2) ? Tt*V(n4,n3)+Area*(Cj0*Vj/(1-M))*
                  (1-exp((1-M)*ln(1-V(n2,Pcathode)/Vj)))
                 : Tt*V(n4,n3)+Area*Cj0*(F1+(1/F2)*(F3*
                   V(n2,Pcathode)-Fc*Vj)+(M/(2*Vj))*
                   V(n2,Pcathode)*V(n2,Pcathode)-Fc*Fc*Vj*Vj))) );
end
endmodule
```

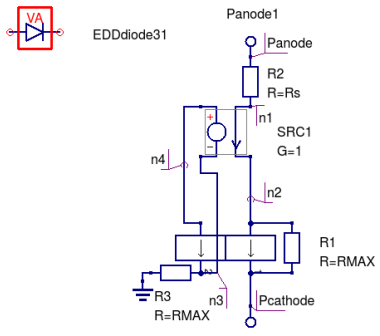
Introduction to the Qucs GPL Verilog-A module synthesizer: Part V

Synthesis of a SPICE like semiconductor diode model: simulated static and dynamic characteristics.



Introduction to the Qucs GPL Verilog-A module synthesizer: Part VI

Verilog-A synthesis of a SPICE like semiconductor diode model: temperature effects




```
Equation
Eqn1
RMAX=1e12
T1=Tnom+273.15
T2=Temp+273.15
Con1=5*N*Vt
Con2=Fc*Vj
Vt=kB*T2/q
F1=(Vj/(1-M))*(1-exp( ( 1-M)*ln(1-Fc)))
F2=exp( (1+M)*ln(1-Fc))
F3=1-Fc*(1+M)
A=7.02e-4
B=1108
Eg_T1=Eg-A*T1*T1/(B+T1)
Eg_T2=Eg-A*T2*T2/(B+T2)
Vj_T2=(T2/T1)*Vj-(2*Vt)*ln(exp(1.5*ln(T2/T1)))-((T2/T1)*Eg_T1-Eg_T2)
Cj0_T2=Cj0*(1+M*(400e-6*(T2/T1)-(Vj_T2-Vj)/Vj))
Is_T2=Is*exp((XTI/N)*ln(T2/T1))*limexp((-Eg_T1/Vt)*(1-T2/T1))
```

```
D1
I1=(V1 > -Con1) ?  Area*Is_T2*(limexp(V1/(N*Vt))-1) : -Is_T2
Q1= (V1 <= Con2) ? Tt*V2+Area*(Cj0_T2*Vj_T2/(1-M))*(1-exp( (1-M)*ln( 1-V1/Vj_T2))) :  Tt*V2+Area*Cj0_T2*(F1+(1/F2)*(F3*(V1-Fc*Vj_T2)+(M/(2*Vj_T2))*(V1*V1-Fc*Fc*Vj_T2*Vj_T2)))
I2=0
Q2=0
```

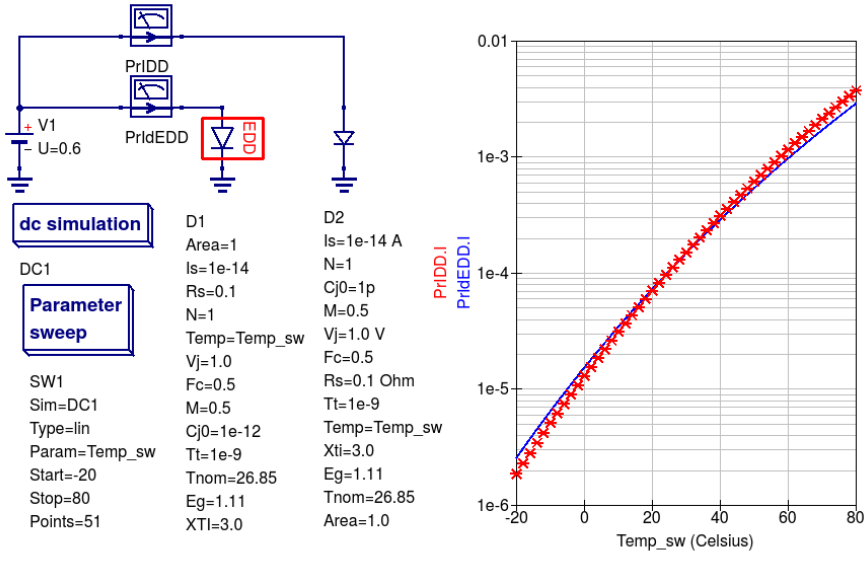
Qucs EDD diode model with temperature effects

Synthesized Verilog-A code

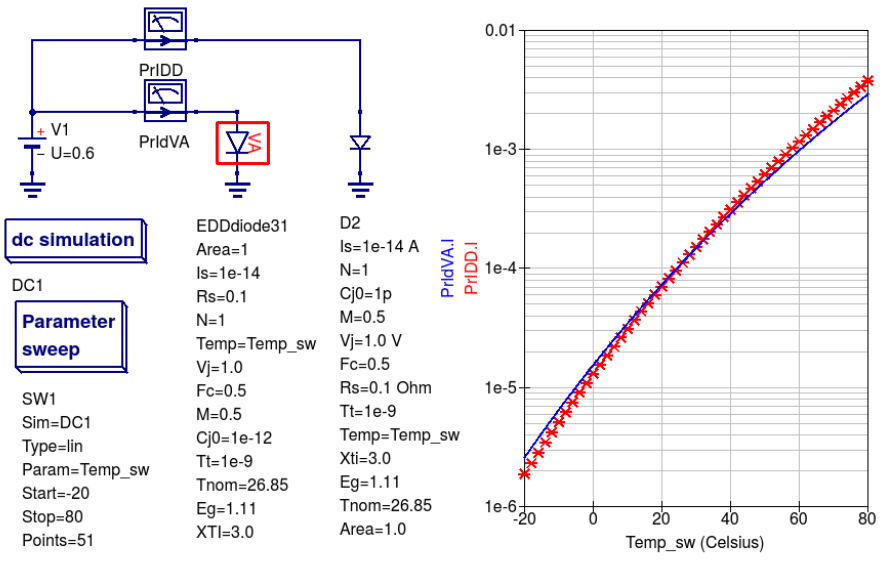
```
`include "disciplines.vams"
`include "constants.vams"
module EDDdiode3(Pcathode, Panode);
inout Pcathode, Panode;
electrical Pcathode, n2, n1, Panode, n4, n3;
parameter real Area=1;   parameter real Is=1e-14;  parameter real Rs=0.1;   parameter real N=1;
parameter real Temp=26.85; parameter real Vj=1.0;  parameter real Fc=0.5;  parameter real M=0.5;
parameter real Cj0=1e-12;    parameter real Tt=1e-9; parameter real Tnom=26.85; parameter real Eg=1.11;
parameter real XTI=3.0;
real RMAX, T1, T2, Con1, Con2, Vt, F1, F2, F3, A, B, EgT1, EgT2, VjT2, Cj0T2, IsT2;
analog begin
@(initial_model)
begin
RMAX=1e12;  T1=Tnom+273.15;  T2=Temp+273.15;  Con1=5*N*Vt;  Con2=Fc*Vj; Vt=`P_K*T2/`P_Q;
F1=(Vj/(1-M))*(1-exp((1-M)*ln(1-Fc))); F2=exp((1+M)*ln(1-Fc)); F3=1-Fc*(1+M); A=7.02e-4; B=1108;
EgT1=Eg-A*T1*T1/(B+T1); EgT2=Eg-A*T2*T2/(B+T2);
VjT2=(T2/T1)*Vj-(2*Vt)*ln(exp(1.5*ln(T2/T1)))-((T2/T1)*EgT1-EgT2);
Cj0T2=Cj0*(1+M*(400e-6*(T2/T1)-(VjT2-Vj)/Vj));
IsT2=Is*exp((XTI/N)*ln(T2/T1))*limexp((-EgT1/Vt)*(1-T2/T1));
end
I(Pcathode,n2) <+ V(Pcathode,n2)/( RMAX );
I(Pcathode,n2) <+ white_noise( 4.0*`PK*( 26.85 + 273.15) / ( RMAX ), "thermal" );
I(n1,Panode) <+ V(n1,Panode)/( Rs );
I(n1,Panode) <+ white_noise( 4.0*`PK*( 26.85 + 273.15) / ( Rs ), "thermal" );
I(n1,n2)  <+  V(n1,n2) * 1e3; I(n3,n4)  <+  I(n3,n4) *1e-3; I(n3,n4)  <+  V(n1,n2) * 1e6 * 1 ;
I(n2,Pcathode) <+ (V(n2,Pcathode)>-Con1)?Area*IsT2*(limexp(V(n2,Pcathode)/(N*Vt))-1):-IsT2;
I(n2,Pcathode) <+ ddt( (V(n2,Pcathode)<=Con2)?Tt*V(n4,n3)+Area*(Cj0T2*VjT2/(1-M))*(1-exp((1-M)*
                       ln(1-V(n2,Pcathode)/VjT2))):
                       Tt*V(n4,n3)+Area*Cj0T2*(F1+(1/F2)*(F3*(V(n2,Pcathode)-Fc*VjT2)+(M/(2*VjT2))*
                       (V(n2,Pcathode)*V(n2,Pcathode)-Fc*Fc*VjT2*VjT2))) );
I(n3) <+ (-V(n3))/( RMAX );
I(n3) <+ white_noise( 4.0*`PK*( 26.85 + 273.15) / ( RMAX ), "thermal" );
end
endmodule
```

Introduction to the Qucs GPL Verilog-A module synthesizer: Part VII

Verilog-A synthesis of a SPICE like semiconductor diode model: simulated $I_d - V_d$ temperature effects.



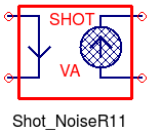
Simulation data for
Qucs EDD model and built-in diode model



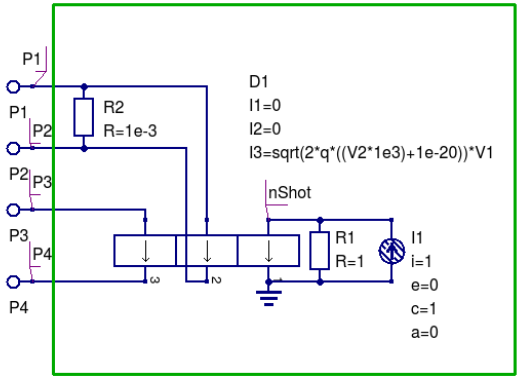
Simulation data for
Verilog-A model and built-in diode model

Introduction to the Qucs GPL Verilog-A module synthesizer: Part VIII

Verilog-A synthesis of semiconductor device shot and flicker noise: EDD models and Verilog-A module code.



Shot_NoiseR11

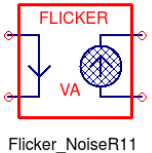


Noise model symbols

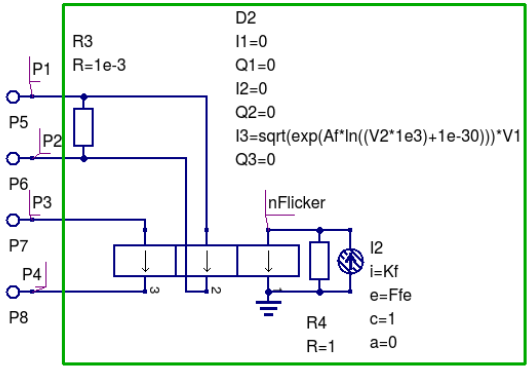
Compact modelling TEMPLATE

```
`include "disciplines.vams"
`include "constants.vams"
module Shot_NoiseR1(P1, P2, P3, P4);
inout P1, P2, P3, P4;
electrical nShot, P2, P1, P3, P4;
analog begin
@(initial_model)
begin
end
I(nShot) <+ (-V(nShot))/( 1 );
I(nShot) <+ white_noise(1,"shot" );
I(P2,P1) <+ V(P2,P1)/( 1e-3 );
I(P3,P4) <+ sqrt(2*`P_Q*((V(P1,P2)*1e3)+1e-20))*V(nShot);
end
endmodule
```

Synthesized Verilog-A module code



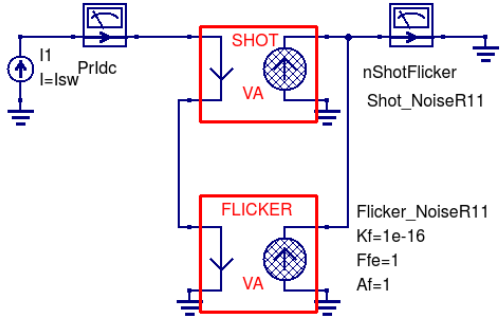
Flicker_NoiseR11
Kf=1e-16
Ffe=1
Af=1




```
`include "disciplines.vams"
`include "constants.vams"
module Flicker_NoiseR1(P1, P2, P3, P4);
inout P1, P2, P3, P4;
electrical P2, P1, nFlicker, P3, P4;
parameter real Kf=1e-12;
parameter real Ffe=1;
parameter real Af=1;
analog begin
@(initial_model)
begin
end
I(P2,P1) <+ V(P2,P1)/( 1e-3 );
I(nFlicker) <+ flicker_noise(Kf, Ffe, "flicker" );
I(nFlicker) <+ (-V(nFlicker))/( 1 );
(P3,P4) <+ sqrt(exp(Af*ln((V(P1,P2)*1e3)+1e-30)))*V(nFlicker);
end
endmodule
```

Introduction to the Qucs GPL Verilog-A module synthesizer: Part IX

Verilog-A synthesis of semiconductor device shot and flicker noise: small signal AC domain simulation data.



ac simulation

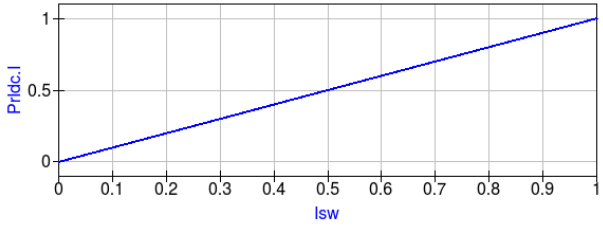
AC1
Type=log
Start=0.01
Stop=100k
Points=141
Noise=yes

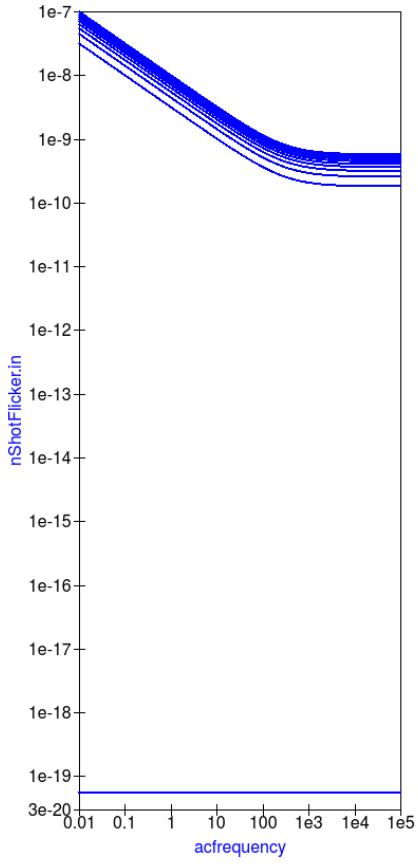
Parameter sweep

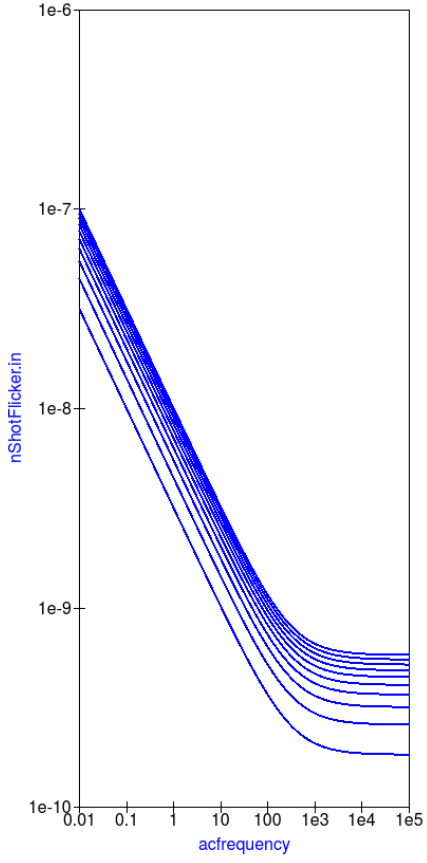
SW1
Sim=AC1
Type=lin
Param=Isw
Start=0
Stop=1
Points=11

dc simulation

DC1



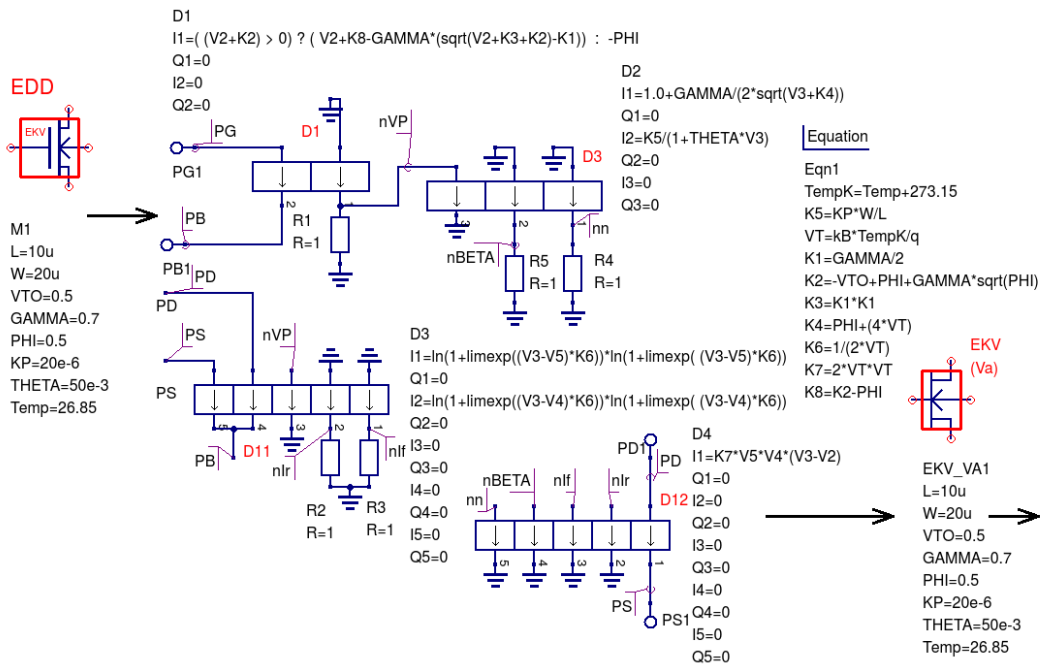




Introduction to the Qucs GPL Verilog-A module synthesizer: Part X

Verilog-A synthesis of multi-EDD models: EKV2p6 nMOS

$I_{ds} = f(V_d, V_g, V_s, V_b)$ model for a transistor operating in long channel mode.

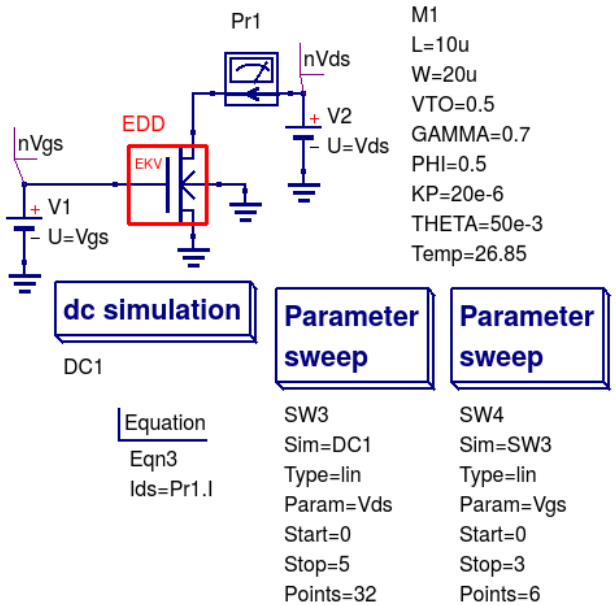


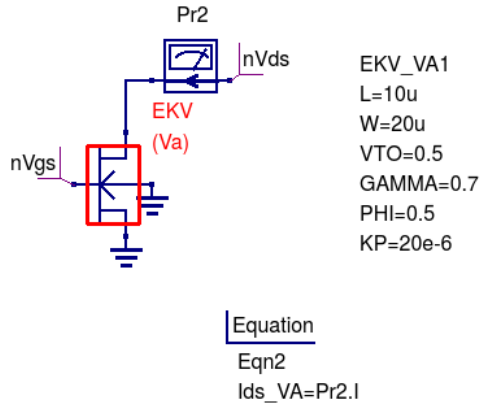
Qucs EDD EKV2p6 Ids=f(Vd, Vg, Vs, Vb) model

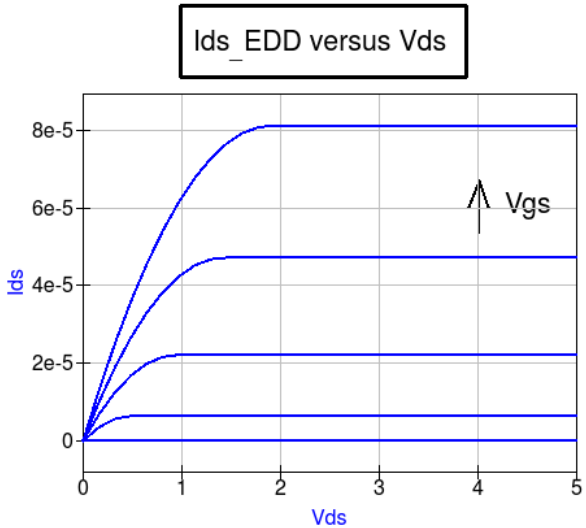
```
`include "disciplines.vams"
`include "constants.vams"
module EKV_VA(PB, PG, PD, PS);
inout PB, PG, PD, PS;
electrical PD, PS, nIr, nIf, nBETA, nn, nVP, PG, PB;
parameter real L=10u;    parameter real W=20u;    parameter real VTO=0.5;
parameter real GAMMA=0.7;    parameter real PHI=0.5;    parameter real KP=20e-6;
parameter real THETA=50e-3;    parameter real Temp=26.85;
real TempK, K5, VT, K1, K2, K3, K4, K6, K7, K8;
analog begin
@(initial_model)
begin
TempK=Temp+273.15;    K5=KP*W/L;    VT=`P_K*TempK/`P_Q;
K1=GAMMA/2;    K2=-VTO+PHI+GAMMA*sqrt(PHI);
K3=K1*K1;    K4=PHI+(4*VT);
K6=1/(2*VT);    K7=2*VT*VT;    K8=K2-PHI;
end
I(PD,PS) <+ K7*V(nn)*V(nBETA)*(V(nIf)-V(nIr));
I(nVP) <+ -(((V(PG,PB)+K2)>0)?(V(PG,PB)+K8-GAMMA*(sqrt(V(PG,PB)+K3+K2)-K1)):-PHI);
I(nVP) <+ V(nVP)/( 1 );
I(nBETA) <+ -((-V(nBETA))/( 1 ));
I(nn) <+ -(1.0+GAMMA/(2*sqrt(V(nVP)+K4)));
I(nBETA) <+ -(K5/(1+THETA*(-V(nVP))));
I(nn) <+ -((-V(nn))/( 1 ));
I(nIr) <+ -((-V(nIr))/( 1 ));
I(nIf) <+ -((-V(nIf))/( 1 ));
I(nIf) <+ -(ln(1+limexp((V(nVP)-V(PS,PB))*K6))*ln(1+limexp((V(nVP)-V(PS,PB))*K6)));
I(nIr) <+ -(ln(1+limexp((V(nVP)-V(PD,PB))*K6))*ln(1+limexp((V(nVP)-V(PD,PB))*K6)));
end
endmodule
```

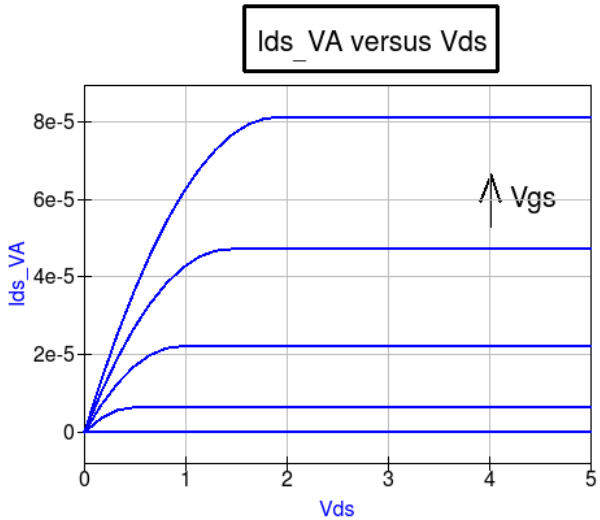
Synthesized EKV2p6 Ids=f(Vd, Vg, Vs, Vb) Verilog-A code

Verilog-A synthesis of multi-EDD models: EKV2p6 nMOS
$I_{ds} = f(V_d, V_g, V_s, V_b)$ swept DC simulation data.









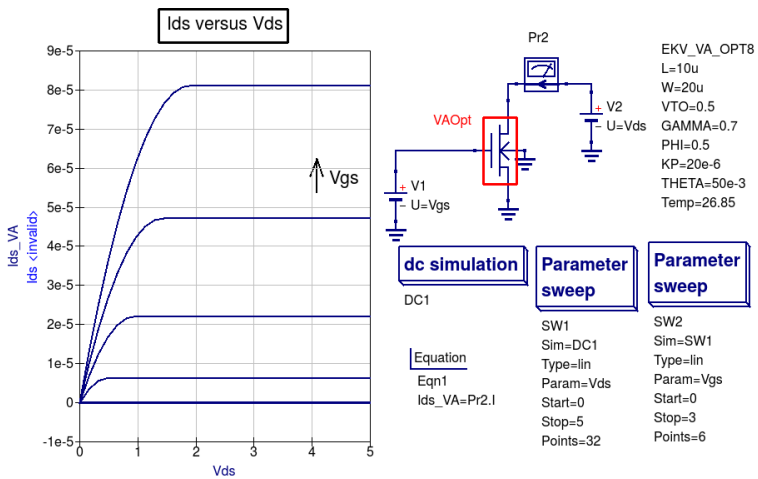
Introduction to the Qucs GPL Verilog-A module synthesizer: Part XII

Verilog-A synthesis of multi-EDD models: Optimization of Qucs synthesized Verilog-A module code for speed.

```
`include "disciplines.vams"
`include "constants.vams"
module EKV_VA_OPT(PB, PG, PD, PS);
inout PB, PG, PD, PS;
electrical PD, PS, PG, PB;
parameter real L=10u;       parameter real W=20u;
parameter real VTO=0.5;   parameter real GAMMA=0.7;
parameter real PHI=0.5;    parameter real KP=20e-6;
parameter real THETA=50e-3;   parameter real Temp=26.85;
real TempK, K5, VT, K1, K2, K3, K4, K6, K7, K8;
real Vg, Vs, Vd, nVP, nBETA, nn, nIf, nIr;
analog begin
@(initial_model)
begin
TempK=Temp+273.15;      K5=KP*W/L;
VT=`P_K*TempK/`P_Q;    K1=GAMMA/2;
K2=-VTO+PHI+GAMMA*sqrt(PHI);   K3=K1*K1;
K4=PHI+(4*VT);    K6=1/(2*VT);    K7=2*VT*VT;
end
Vg   = V(PG,PB);   Vs   = V(PS,PB);   Vd   = V(PD,PB);
nVP   = (Vg+K2>0)?(Vg+K2-PHI-GAMMA*(sqrt(Vg+K2+K3)-K1)):-PHI;
nBETA = K5/(1+THETA*nVP);
nn    = 1.0+GAMMA/(2*sqrt(nVP+K4));
nIf   = ln(1+limexp((nVP-Vs)*K6))*ln(1+limexp((nVP-Vs)*K6));
nIr   = ln(1+limexp((nVP-Vd)*K6))*ln(1+limexp((nVP-Vd)*K6));
I(PD,PS) <+ K7*nn*nBETA*(nIf-nIr);
end
endmodule
```

EKV2p6 EKV_VA_OPT optimized Verilog-A module code

TEST MODULE



NOTES

1. At this stage in the development of the Qucs synthesizer optimized Verilog-A module code is done manually.
2. General procedure:
   - 2.1 Reduce current contribution statements to a minimum. This can be done by representing model equation quantities by real variables rather than internal node voltages.
     [one I(a) <+ ..... in the EKV nMOS example]
   - 2.2 Eliminate as many as possible internal model nodes and remove current to voltage one Ohm conversion resistors.
     [zero left in EKV nMOS example]

A comment on the Qucs simulation process:

Simple simulation run time tests indicate that the optimized EKV2p6 Verilog-A model simulation speed is at least 30X faster than the interactive EDD model.

- End of Part 1